Visual Basic (VB) Programming courses provide an opportunity for students to gain expertise in computer programs using the Visual Basic (VB) language. As with more general computer programming courses, the emphasis is on how to structure and document computer programs and how to use problem-solving techniques. These courses cover such topics as the use of text boxes, scroll bars, menus, buttons, and Windows applications. More advanced topics may include mathematical and business functions and graphics.
Visual Basic Computer Programming

Instructor Information
Instructor: [Instructor Information]
Instructor Email: [Instructor Email]
Instructor Website: [Instructor Website]
Room: [Room]
Period: [Period]
Grade Level: 10-12
Course Prerequisites: Introduction to Information Technology, Computer Applications and Algebra 1

Course Description

Visual Basic is designed to provide the introduction to intermediate working knowledge of Visual Basic and the skills needed for software development. Students will learn how to create user interfaces as well as the coding behind the interface.

Emphasis is place on the three-step process of building an application, creating the user interface, setting properties, writing code and solving problems.

Students enrolled in Visual Basic should be able to understand story problems and create solutions to problems.

Visual Basic computer programming is one-trimester course for .5 credit. The course is designed for students interested in a career in software development, information systems, or business.

Prerequisite for the course is ½-computer credit and a strong mathematical background.

Instructional Philosophy

Students will complete challenging programming labs in which they will learn and demonstrate knowledge of programming skills. High quality work is expected, and students will be given opportunity to redo work until it meets industry standards given during instruction. Classroom activities include course reading, programming labs, and problem-solving activities, end of chapter questions and course quizzes and exams. On some projects, students will work in teams but will be expected to complete individual assignments in relation to the team’s project. Assessment methods will include written exams, tests, quizzes, projects, oral and written research reports.

South Dakota 9-12 Educational Technology Content Standards

Visual Basic is an advanced programming course designed for students in grades 10 through 12. It is important that the course not only address the Course Standards, but also the South Dakota Educational Technology Standards. This course is should meet the advanced level of the South Dakota Educational Technology Standards.
South Dakota 9-12 Educational Technology Content Standards

Students understand the history and progression of technology in relation to the development and design of future technology
9-12.NC.1.1 Compare and contrast how societal changes mirror innovations and emerging technologies.
9-12.NC.1.2 Predict how the evolution of technology will influence the design and development of future technology.

Students analyze the parts of a technological system in terms of input, process, output, feedback.
9-12.NC.2.1 Analyze technology systems to make informed choices

Students analyze the relationships and the connections between technologies in different fields of study and how they apply to communities.
9-12.NC.3.1 Analyze intended and unintended impacts of a system
9-12.NC.3.2 Integrate technology into school, home and community
9-12.NC.3.3 Evaluate technologies that increase educational and workplace opportunities

Students understand the purpose and demonstrate the use of the design process in problem solving.
9-12.NC.4.1 Compare and contrast other problem-solving and decision-making methods
9-12.NC.4.2 Formulate a technological solution using data-driven decision-making.

Students understand the safe, ethical, legal, and societal issues related to technology.
9-12.SI.1.1 Evaluate the need for acceptable use policies
9-12.SI.1.2 Compile a list of immediate and long-range effects of ethical and unethical uses of technology on individuals and society.

Students investigate the advantages and disadvantages of technology.
9-12.SI.2.1 Analyze advantages and disadvantages of widespread use and reliance on technology in the workplace and in society as a whole.
9-12.SI.2.2 Compare and contrast society’s influence on technology and technology’s influence on society.

Students recognize and demonstrate skills in operating technological systems.
9-12.CT.1.1 Incorporate knowledge and enhanced usage skills to create a product.
9-12.CT.1.2 Apply strategies for identifying and solving routine hardware and software issues.

Students use technology to enhance learning, extend capability, and promote creativity.
9-12.CT.2.1 Utilize a virtual learning environment as a strategy to build 21st century learning skills
9-12.CT.2.2 Investigate to apply expert systems, intelligent agents, and simulations in real-world situations
9-12.CT.2.3 Utilize online information resources routinely and efficiently to meet needs for collaboration, research, publication, communication, and productivity.

Students evaluate and select information tools based on the appropriateness to specific tasks.
9-12.CT.3.1 Select and apply technology tools for research, information analysis, problem solving, and decision making in content learning.
9-12.CT.3.2 Organize and manage personal/professional information using technology tools. (e.g., finances, schedules, addresses, purchases, correspondences).

Students understand the purpose of information technologies to communicate with a variety of collaborators.
9-12.CP.1.1 Collaborate with external peers, experts, and others by using technology to compile, synthesize, produce, and disseminate information, models, and other creative works.

Students exchange information and ideas for an identified purpose through Information technologies.
9-12.CP.2.1 Adapt delivery of communication based on available information technologies.

Students use technology to locate and acquire information.
9-12.IL.1.1 Design a research project using a variety of technologies to find information to solve real world problems.
Students determine the reliability and relevancy of information.
9-12.II.2.1 Independently evaluates the accuracy, relevance, appropriateness, comprehensiveness, and bias of electronic information services.

Course Standards

**Students will demonstrate programming as it relates to the customer needs**
1.1 Students will gather data to identify customer requirements.
1.2 Students will demonstrate knowledge of programming language concepts.
1.3 Students will develop software requirements specification.

**Students will produce IT-based strategies and project plans to solve the problem.**
2.1 Students will define scope of work for the programming project.
2.2 Students will demonstrate knowledge and skills of working in a software development team.

**Students will demonstrate knowledge of the software development process.**
3.1 Students will demonstrate knowledge of software development methodology.
3.2 Students will apply tools for developing software applications.
3.3 Students will apply language specific programming tools/techniques.

**Students will create a logical design for a software application.**
4.1 Students will create design specification for a computer application.
4.2 Students will analyze real world problems for the applicability of structured, object orientate, even driven logical design methods.

**Students will create a computer application by writing code.**
5.1 Students will demonstrate knowledge of programming language concepts.
5.2 Students will develop an application using selected programming language.
5.3 Students will demonstrate knowledge of basic software systems implementation.
Programming 1 - Major Projects and Assignments

(These projects may change at the discretion of the instruction during the course of the semester)

Programming Careers: Students will research computer programming careers and the colleges and majors that will help them to obtain a job as a computer programmer. Students will write and demonstrate their findings on the best schools and careers that are available for people that are interested in computer science and visual basic programming.

Introduction to Visual Basic: Students will begin creating new Visual Basic projects that will include changing form properties such as the caption, name, and backcolor property. Students will begin adding controls to the form as well as adding simple code to make the form interactive. Through this section, students will work on creating a variety of programs, debugging, and answering end of unit questions.

Designing and Application: Students will understand terminology and planning for an Object Orientated Event Driven application. This will include identifying the applications tasks, objects, events and drawing the user interface. After the plan phase is over students will begin creating the interface and then will code, test, debug and document the program. Through this section, students will work on creating a variety of programs, debugging, and answering end of unit questions.

Using Variables and Constants: Students will begin working with variables. While working with variables students will understand data types, naming variables, declaring variables, storing data, and the scope of variables. Emphasis will be placed on local, form level and global variables. Students will begin working with built-in visual basic functions to enhance and add functionality to the user interface. Through this section, students will work on creating a variety of programs, debugging, and answering end of unit questions.

Selection Structure: Students will begin using selection structure statements such as the if…then…else and case select statements. Other items that will be covered to assist in the use of selection statements include relational operators, UCase functions, check box controls, call statements, msgbox function, GotFocus Event, pseudocode, and flowcharting. Through this section, students will work on creating a variety of programs, debugging, and answering end of unit questions.

Repetition Structure: Students will introduce the use of the repetition structure into their programs. Using the repetition structure students will utilize the use of the for next, do while, and do until loops. Students will also use control arrays and the change event to increase the functions of the program and to utilize the repetition structure of programming. Through this section, students will work on creating a variety of programs, debugging, and answering end of unit questions.

Sequential Access Files, Menus, and Reports: Students will incorporate the use of list boxes to read and write information to data files. Students will be introduced to the end of file function, print method, menus, clipboard objects, and string manipulation functions. Through this section, students will work on creating a variety of programs, debugging, and answering end of unit questions.
Programming 2 - Major Projects and Assignments

(These projects may change at the discretion of the instruction during the course of the semester)

Dialog Boxes and Error Trapping: Students will create programs that use dialog boxes for user interaction. With the dialog boxes, the programs will have to incorporate the use of the filter and filename property. Programs will also include Input and LOF functions. Through this section, students will work on creating a variety of programs, debugging, and answering end of unit questions.

Random Access File: Students will create programs and compare the difference between random and sequential files and the storage required for both styles of access. Incorporated into the programs will be the concept of passing information to a sub procedure, creating user-defined procedures to receive data, and coding other components of the form. Through this section, students will work on creating a variety of programs, debugging, and answering end of unit questions.

Database Access: Students will learn database concepts and terminology in which they can use data validation rules to utilize the information within new and existing databases. Programs will utilize the ADO data controls and the SQL select command. Through this section, students will work on creating a variety of programs, debugging, and answering end of unit questions.

Variable Arrays: Student programs will include the use of one-dimensional variable arrays. With the use of arrays, students will look at the advantages and storing of data. Variable arrays will also include the use of the data grid control and invoking a function procedure. Through this section, students will work on creating a variety of programs, debugging, and answering end of unit questions.

Difference between Visual Basic Versions: Students will identify the difference between Visual Basic 6.0 and Visual Basic.Net. Students will explore the Visual Studio.net environment, the solution explorer, using the Windows form, changing the appearance of a form, and using the Active X controls. Through this section, students will work on creating a variety of programs, debugging, and answering end of unit questions.

Assessment Plan

All assignments are designed to show whether students have met the standards for the course. Any lesson questions, programming labs, quizzes, and tests that are assessed as “poor quality” will be expected to be REDONE for higher credit.

Distribution of Grading Components

Grades are determined by dividing the points earned by the total number of points available in the grading period. Each major project and assignment commands an approximate percentage of the total points for the grading period as follows:

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reading Assignment/Daily Grades</td>
<td>25%</td>
</tr>
<tr>
<td>Programming Assignments</td>
<td>45%</td>
</tr>
<tr>
<td>Tests, Quizzes, and Exams</td>
<td>20%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>10%</td>
</tr>
</tbody>
</table>
## Description of Grading and Quality Work

<table>
<thead>
<tr>
<th>Grade</th>
<th>Scale</th>
<th>Description of Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>94-100%</td>
<td>Consistently demonstrates an exceptional level of quality and effort. Having all work in on time and completed to exceed expectations. Mastery in evaluating, synthesizing, and applying the knowledge and skills of information technology.</td>
</tr>
<tr>
<td>B</td>
<td>87-93.9%</td>
<td>Consistently demonstrates proficient knowledge with a good effort and quality of work. All assignments are complete and on time. Demonstrates the ability to evaluate, analyze, synthesize and apply the principles of information technology.</td>
</tr>
<tr>
<td>C</td>
<td>78-86.9%</td>
<td>Demonstrates proficient knowledge and the ability to apply information technology. Work shows average effort. A few assignments may be missed or late.</td>
</tr>
<tr>
<td>D</td>
<td>68-77.9%</td>
<td>Work shows minimal effort and some assignments are late. Demonstrates a basic understanding of recalling or comprehending information technology.</td>
</tr>
<tr>
<td>F</td>
<td>Below 67.9%</td>
<td>Understanding is below basic in relation to information technology. Work is of poor quality and does not meet standards or expectations.</td>
</tr>
</tbody>
</table>

## Extra Help
Extra help is available from 8-8:15 am and after school until 4 pm. Students may also sign out to come in for help during SRP.

## Course Textbook
Computer Programming

Rationale Statement:
The world is full of problems that need to be solved or that need a program to solve them faster. In computer, programming students will learn how to solve story problems and develop a computer program that will solve the problem. Computer programming courses in the state of South Dakota usually are taught using one of three computer languages: Visual Basic, C++, or Java. Students that are interested in the Programming and Software Development pathway will find that taking a computer-programming course in high school will better prepare them for post-secondary work in computer science, engineering, mathematics, and other software development areas.

Course Description:
Grade Level:  10-12

Course Topics:

- Introduction to programming history and the programming language
- Understanding the information processing cycle
- Customer needs analysis for designing a program
- Defining and designing the program project
- Coding an application
- Creating, debugging, and documenting a software application

NOTE: The core technical standards and examples are designed for a Programming 1 and Programming 2 Course.
## Indicator #1: Students will demonstrate programming as it relates to the customer needs

<table>
<thead>
<tr>
<th>Bloom’s Taxonomy Level</th>
<th>Standard and Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analyzing</strong></td>
<td><strong>1.1 Students will gather data to identify customer requirements.</strong></td>
</tr>
<tr>
<td></td>
<td>For example, to meet this standard students may:</td>
</tr>
<tr>
<td></td>
<td>- Gather information using interviewing strategies.</td>
</tr>
<tr>
<td></td>
<td>- Identify input, output and system processing requirements.</td>
</tr>
<tr>
<td></td>
<td>- Clarify specifications using questioning techniques.</td>
</tr>
<tr>
<td></td>
<td>- Identify hardware, networking, and software system functional requirements.</td>
</tr>
<tr>
<td></td>
<td>- Demonstrate knowledge of nonfunctional requirements.</td>
</tr>
<tr>
<td></td>
<td>- Use customer satisfaction in determining product characteristics.</td>
</tr>
<tr>
<td><strong>Applying</strong></td>
<td><strong>1.2 Students will demonstrate knowledge of programming language concepts.</strong></td>
</tr>
<tr>
<td></td>
<td>For example, to meet this standard students may:</td>
</tr>
<tr>
<td></td>
<td>- Demonstrate knowledge of the concept of physical representation of digitized information.</td>
</tr>
<tr>
<td></td>
<td>- Demonstrate knowledge of the hardware-software connection.</td>
</tr>
<tr>
<td></td>
<td>- Demonstrate knowledge of the function and operation of compilers and interpreters.</td>
</tr>
<tr>
<td></td>
<td>- Demonstrate knowledge of current key programming languages and the environment they are used in.</td>
</tr>
<tr>
<td><strong>Evaluating</strong></td>
<td><strong>1.3 Students will develop software requirements specification.</strong></td>
</tr>
<tr>
<td></td>
<td>For example, to meet this standard students may:</td>
</tr>
<tr>
<td></td>
<td>- Demonstrate knowledge of the use, structure, and contents of a requirements specification document.</td>
</tr>
<tr>
<td></td>
<td>- Define system and software requirements.</td>
</tr>
<tr>
<td></td>
<td>- Define business problem to be solved by the application</td>
</tr>
<tr>
<td></td>
<td>- Develop informal specifications.</td>
</tr>
<tr>
<td></td>
<td>- Develop formal specification.</td>
</tr>
<tr>
<td></td>
<td>- Review and verify specification with customer.</td>
</tr>
</tbody>
</table>
Indicator #2: Students will produce IT-based strategies and project plans to solve the problem.

<table>
<thead>
<tr>
<th>Bloom’s Taxonomy Level</th>
<th>Standard and Examples</th>
</tr>
</thead>
</table>
| **Understanding**      | 2.1 *Students will define scope of work for the programming project.*  
                          For example, to meet this standard students may:  
                          • Demonstrate knowledge of the key functions and subsystems of the software product.  
                          • Demonstrate knowledge of software development process and issues.  
                          • Develop implementation plan. |
| **Applying**           | 2.2 *Students will demonstrate knowledge and skills of working in a software development team.*  
                          For example, to meet this standard students may:  
                          • Identify resources and risks.  
                          • Demonstrate knowledge of cross-functional team structures and team members’ roles. |
**Indicator #3: Students will demonstrate knowledge of the software development process.**

<table>
<thead>
<tr>
<th>Bloom’s Taxonomy Level</th>
<th>Standard and Examples</th>
</tr>
</thead>
</table>
| **Understanding**       | 3.1 Students will demonstrate knowledge of software development methodology.  
For example, to meet this standard students may:  
  - Demonstrate knowledge of system analysis issues related to design, testing, implementation, and maintenance.  
  - Identify roles on team members/customers in the software development process.  
  - Demonstrate knowledge of how to use software methodologies to analyze a real-world problem.  
  - Identify constraints of the current project.  
  - Demonstrate knowledge of modeling and analyzing functional requirements (e.g., dataflow diagrams, process specifications, and a data dictionary).  
  - Demonstrate knowledge of modeling and analyzing data requirements (e.g., Jackson diagrams, entity relationship diagrams, and relations). |
| **Applying**            | 3.2 Students will apply tools for developing software applications.  
For example, to meet this standard students may:  
  - Demonstrate knowledge of software development environment.  
  - Use prototyping techniques.  
  - Use desk checking  
  - Demonstrate knowledge of reuse and components. |
| **Applying**            | 3.3 Students will apply language specific programming tools/techniques.  
For example, to meet this standard students may:  
  - Develop programs using appropriate language.  
  - Make use of appropriate development environment for the selected language. |
**Indicator #4: Students will create a logical design for a software application.**

<table>
<thead>
<tr>
<th>Bloom’s Taxonomy Level</th>
<th>Standard and Examples</th>
</tr>
</thead>
</table>
| **Evaluating**          | **4.1 Students will create design specification for a computer application.**  
  For example, to meet this standard students may:  
  - Analyze real world problems for the applicability of structured, object oriented, event-driven logical design methods.  
  - Design system input, output, processing, and interfaces. |
| **Applying**            | **4.2 Students will analyze real world problems for the applicability of structured, object orientate, even driven logical design methods.**  
  For example, to meet this standard students may:  
  - Demonstrate knowledge of the characteristics and the uses of processing  
  - Identify basic concepts of algorithm and data structure development.  
  - Demonstrate knowledge of different data types  
  - Identify constraints.  
  - Demonstrate knowledge of nonfunctional requirements  
  - Demonstrate knowledge of modular design concepts.  
  - Demonstrate knowledge of how to design and implement programs in a top-down manner.  
  - Analyze and prepare logic using program flowchart.  
  - Identify standards and issues related to I/O programming and design of I/O interfaces. |
Indicator #5: Students will create a computer application by writing code.

<table>
<thead>
<tr>
<th>Bloom’s Taxonomy Level</th>
<th>Standard and Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Applying</strong></td>
<td>5.1 Students will demonstrate knowledge of programming language concepts.</td>
</tr>
<tr>
<td></td>
<td>For example, to meet this standard students may:</td>
</tr>
<tr>
<td></td>
<td>• Demonstrate knowledge of the basics of structured, object-oriented, and event-driven</td>
</tr>
<tr>
<td></td>
<td>programming.</td>
</tr>
<tr>
<td></td>
<td>• Demonstrate knowledge of the concepts of data and procedural representation.</td>
</tr>
<tr>
<td><strong>Applying</strong></td>
<td>5.2 Students will develop an application using selected programming language.</td>
</tr>
<tr>
<td></td>
<td>For example, to meet this standard students may:</td>
</tr>
<tr>
<td></td>
<td>• Translate logical design into code in an appropriate language argument.</td>
</tr>
<tr>
<td></td>
<td>• Demonstrate knowledge of specific language syntax.</td>
</tr>
<tr>
<td><strong>Evaluating</strong></td>
<td>5.3 Students will demonstrate knowledge of basic software systems implementation.</td>
</tr>
<tr>
<td></td>
<td>For example, to meet this standard students may:</td>
</tr>
<tr>
<td></td>
<td>• Compile and debug code.</td>
</tr>
<tr>
<td></td>
<td>• Prepare code documentation.</td>
</tr>
<tr>
<td></td>
<td>• Conduct code walkthrough and/or inspection.</td>
</tr>
<tr>
<td></td>
<td>• Troubleshoot unexpected results.</td>
</tr>
<tr>
<td></td>
<td>• Access needed information using company and manufacturers' references.</td>
</tr>
</tbody>
</table>
**Stage 1: Desired Results**

Core Standard(s): Indicator #1: Students will demonstrate programming as it relates to the customer needs

1.1 Students will gather data to identify customer requirements.

**Understandings:** Students will understand that….

When developing a program it is important to make sure that you are meeting the requirements set by our customer. You need to take into consideration all the items they want to have input, what and how they want the output to look like, and what they want the program to do. With these simple items it is easy to begin creating your program with your customer in mind.

Think of the creation of your program as a solution to a standard story problem. When we look at the story problem we can start to get an idea of how to generate the program.

When gathering the requirements for the program it is important to completely understand the problem and data needed to solve it, breakdown the problem in English sentences or pseudocode, sketch a picture of the interface, and then identify the tasks.

We want our program to be accurate, easy to understand, easy to change, and efficient.

**Essential Question(s):**
What is the problem we are trying to solve?
What information do we need from the user?
What do we need to do with the information?
What calculations or tasks are needed?
What will the output look like?
How do we know that the program is working correctly?

**Students will know…:**
How to identify the problem
How to identify the input needs
How to determine tasks
How to display the output

**Student will be able to …:**
Write pseudocode before coding
Sketch the user interface
Explain how they will test to see if the output is correct

---

**Stage 2: Assessment Evidence**

What evidence will show that students understand?

- [X] Performance Task
- [X] Project
- [X] Tests
- [X] Informal Observations
- [X] Interviews
- [X] Self-Assessment
- [X] Quizzes
- [X] Discussions
- [X] Other
### Stage 3: Learning Plan

<table>
<thead>
<tr>
<th>Motivation – Introduce and Explain</th>
</tr>
</thead>
<tbody>
<tr>
<td>How will you help students know <em>where</em> they are headed and why?</td>
</tr>
<tr>
<td>How will you hook students through engaging and thought-provoking experiences that point toward big ideas, essential questions, and performance tasks?</td>
</tr>
</tbody>
</table>
Model (Teacher presentation):
What instruction is needed to *equip* students for final performance?

Let’s think about a program and what we need to do to create it. As we know the first stage of the development life cycle is to analyze the requirements. Let’s start with something where the requirements are pretty easy to determine.

Cosmo Spacely runs a small manufacturing company. To help with his payroll he would like to have a program that would help him calculate each employees weekly salary. An employee receives time and half for every hour they work over 40 hours. Mr. Spacely would like the program to display the employee name, employee number, number of regular hours, regular pay, employee wage, number of overtime hours, overtime pay, total hours, and total pay. Mr. Spacely just wants to have to type in the employee name, number, employee wage, and hours worked.

What information does Mr. Spacely need to put in the program?
What processes have to take place?
What information needs to be displayed?

After we have the requirements decided, let’s design the program by using a flowchart.

Before we design our user interface, let’s put together a TOE chart. What we have done with our flowchart will help out with the TOE chart, and what we discover in the TOE chart will help us design a user interface.

After the program has been planned the teacher will draw up the user interface with the assistance of the students. This is a great time to discuss usability and aesthetics.
Guided and Independent Practice (Student Engagement):
What events can students experience to make the ideas and issues real? What learning activities will help student to explore the big ideas and essential questions?

- Engage students in the discussion of program needs and design.
- Engage students in a discussion about how are the many different ways to calculate the information.
- Have students create a flowchart for a story problem. Have students create a TOE chart for a story problem.
- Have students sketch two different user interfaces for the program.

Reflection/Assessment:
How will you cause students to reflect and rethink to dig deeper into core ideas?
How will you guide students in rehearsing, revising, and refining their work based on feedback and self-assessment?
How will students exhibit their understanding about their final performances and products?
How will you guide them in self-evaluation to identify the strengths and weaknesses in their work and set future goals?

The planning of a program can save a programmer a lot of time. It is important that we first break down what the program needs to do and then what information we need to gather. Using flowcharts and TOE charts are a great way to take the information and prepare it before you create your program.

Were there any steps that you could have eliminated? Any information that would have been easier to calculate? Did you see any other ways to calculate your information? How could you improve the process and look of your design.
Subject/Course: Computer Programming  
Grade(s): 9 – 12  
Designer(s) Rob Honomichl

### Stage 1: Desired Results

<table>
<thead>
<tr>
<th>Core Standard(s):</th>
<th>Indicator #3: Students will demonstrate knowledge of the software development process.</th>
</tr>
</thead>
</table>

3.1 Students will demonstrate knowledge of software development methodology.

**Understandings:** Students will understand that….

When developing software, it is important to follow a structure to develop the software product. Sometimes this process is known as the software life cycle or the software process. The software development process consists of a variety of tasks or activities that take place during the process. These activities help to plan and create the software tasks.

When developing software development activities include analyzing the requirements, designing, implementation, testing. In the general model each step produces deliverables required by the next phase in the life cycle. In the waterfall model, each phase must be completed in its entirety before the next phase can begin. Other models that are less common are the v-shape model, the incremental model, and spiral model. All of these models require the same steps it is just the way that you go from one step to the next that differs.

With the software development process there are also very important tools that programmers use to plan out their program. These tools include flowcharts and task, object, event charts. These tools help with the planning process of the program. These tools help you to develop the process, requirements, and documentation of your program.

**Essential Question(s):**

Define the requirements of the program you are creating?  
Define the information you will need for the program?  
What processes have to take place?  
How are you going to represent the flow of the program?

Students will know….

<table>
<thead>
<tr>
<th>How to define the requirements (Stage 1) of the program application.</th>
<th>Student will be able to …..</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to define the tasks (processes), objects, and events needed in the program application.</td>
<td>Define the requirements of the program application</td>
</tr>
<tr>
<td>How to design the user interface.</td>
<td>Create the user interface with appropriate objects to gather and display information.</td>
</tr>
<tr>
<td>How to represent the program by flowcharting and using a TOE chart.</td>
<td>Flowchart the program</td>
</tr>
</tbody>
</table>

### Stage 2: Assessment Evidence

What evidence will show that students understand?  

<table>
<thead>
<tr>
<th>X Performance Task</th>
<th>X Project</th>
<th>X Quizzes</th>
</tr>
</thead>
<tbody>
<tr>
<td>X Tests</td>
<td>X Informal Observations</td>
<td>X Discussions</td>
</tr>
<tr>
<td>X Interviews</td>
<td>X Self-Assessment</td>
<td>X Other</td>
</tr>
</tbody>
</table>

---

Software Development Life Cycle (Part 1&2)
## Stage 3: Learning Plan

<table>
<thead>
<tr>
<th>Motivation – Introduce and Explain</th>
</tr>
</thead>
<tbody>
<tr>
<td>How will you help students know <em>where</em> they are headed and why?</td>
</tr>
<tr>
<td>How will you <em>hook</em> students through engaging and thought-provoking experiences that point toward big ideas, essential questions, and performance tasks?</td>
</tr>
</tbody>
</table>

When you get ready to program you can’t just hop on the computer and start typing. It takes some planning and some time to think about the problem and what you are going to do. The development of software comes in a multiple steps from analyzing the requirements, designing, implementation, and then testing. After the last phase the cycle may start over or you slowly work back through it.

There are different models that help with the cycle of programming. The two most common ones are the general model and the waterfall model. These models are very similar. Another key part of the planning phase is using tools like flowcharting and a Task, Object, Event (TOE) chart to plan your program. These tools will help you to think about the information you need to gather, what needs to happen once you have it, and then what you are going to do with it once it has been calculated.

At this point we are going to look at the first two steps of the cycle. We need to find the requirements and then we need plan the design of the program using a flowchart and a TOE chart.
Model (Teacher presentation):
What instruction is needed to *equip* students for final performance?

Let’s think about a program and what we need to do to create it. As we know the first stage of the development life cycle is to analyze the requirements. Let’s start with something where the requirements are pretty easy to determine.

Cosmo Spacely runs a small manufacturing company. To help with his payroll he would like to have a program that would help him calculate each employees weekly salary. An employee receives time and half for every hour they work over 40 hours. Mr. Spacely would like the program to display the employee name, employee number, number of regular hours, regular pay, employee wage, number of overtime hours, overtime pay, total hours, and total pay. Mr. Spacely just wants to have to type in the employee name, number, employee wage, and hours worked.

What information does Mr. Spacely need to put in the program?
What processes have to take place?
What information needs to be displayed?

After we have the requirements decided, let’s design the program by using a flowchart.

Before we design our user interface, let’s put together a TOE chart. What we have done with our flowchart will help out with the TOE chart, and what we discover in the TOE chart will help us design a user interface.

After the program has been planned the teacher will draw up the user interface with the assistance of the students. This is a great time to discuss usability and aesthetics.
Guided and Independent Practice (Student Engagement):
What events can students experience to make the ideas and issues real? What learning activities will help student to explore the big ideas and essential questions?

- Engage students in the discussion of program needs and design.
- Engage students in a discussion about how are the many different ways to calculate the information.
- Have students create a flowchart for a story problem. Have students create a TOE chart for a story problem.
- Have students sketch two different user interfaces for the program.

Reflection/Assessment:
How will you cause students to reflect and rethink to dig deeper into core ideas?
How will you guide students in rehearsing, revising, and refining their work based on feedback and self-assessment?
How will students exhibit their understanding about their final performances and products?
How will you guide them in self-evaluation to identify the strengths and weaknesses in their work and set future goals?

The planning of a program can save a programmer a lot of time. It is important that we first break down what the program needs to do and then what information we need to gather. Using flowcharts and TOE charts are a great way to take the information and prepare it before you create your program.

Were there any steps that you could have eliminated? Any information that would have been easier to calculate? Did you see any other ways to calculate your information? How could you improve the process and look of your design.
Designing a Program

Subject/Course: Computer Programming
Grade (s): 9 – 12  Designer (s) Rob Honomichl

### Stage 1: Desired Results

**Core Standard(s):** Students will create a computer application by writing code.

5.2 Students will develop an application using selected programming language.

**Understandings:** Students will understand that….
A program is just an application that solves a story problem. That story problem could be as simple as displaying the answer to a simple equation, or as complex as what a costumer may need for their business. But before you can just jump into the program you have to do some background work. As we look at planning out a program it is important to understand the design methodologies surrounding program design.

As we prepare to design a program it is important to remember that the program must be accurate, easy to understand, easy to change, and efficiently executed. In Visual Basic, not only do we need to be concerned about the code, but also the objects that will execute the code.

**Essential Question(s):**
What is the problem we are going to solve with this application?
What input do we need from the user, and what are we going to use to get it?
What and how are we going to display the information the user is requesting?
What equations do we need to know?
What variables do we need to declare?
What other requirements are necessary to know?

**Students will know….**
- How to select the appropriate objects to meet the requirements.
- How to declare and initialize variables.
- How to design a user-friendly interface.
- How to use the appropriate syntax to create an efficient and accurate application.

**Student will be able to …..**
- Create the user interface with appropriate objects to gather and display information.
- Declare and use variables to execute the application more efficiently.
- Code the application to be accurate and efficient.
- Meet the specifications and requirements to solve the problem.

### Stage 2: Assessment Evidence

What evidence will show that students understand?
- [ ] Performance Task
- [ ] Project
- [x] Tests
- [x] Informal Observations
- [ ]Quizzes
- [ ]Discussions
- [ ]Self-Assessment
- [ ]Other
### Motivation – Introduce and Explain

How will you help students know where they are headed and why? How will you *hook* students through engaging and thought-provoking experiences that point toward big ideas, essential questions, and performance tasks?

The world is full of problems, and through programming we can create applications to solve the problems of our customers. The best way to think about any program that you need to create is to think of it as a story problem. We need to think of the input, process, and the output the problem. For example, what if you were working for a company that wanted an application to print out information for building permits. We need to first ask the customer what type of information they need on their permits. Once we have that information we can begin creating an application to solve this problem.

In programming there are different ways to plan out how this information can be laid out. Your planning phase can save you a lot of time in the long run. In this application I will give you the choice to either create a flowchart or a Task, Object, Event (TOE) chart to plan our your program. These tools will help you to think about the information you need to gather, what needs to happen once you have it, and then what you are going to do with it once it has been calculated.

The best part about creating programs is you can use them for work and for play. Another thing that we will look into is creating them for more devices than just you desktop or laptop.

### Model (Teacher presentation):

What instruction is needed to *equip* students for final performance?

To demonstrate to the student the planning phase the teacher will put up a simple story problem. The discussion will then be centered on what input, process, and output need to be considered to solve the problem.

With that information, a demonstration of flowcharting and creating a TOE chart will be done with this information.

After the program has been planned the teacher will draw up the user interface with the assistance of the students. This is a great time to discuss usability and aesthetics.
Guided and Independent Practice (Student Engagement):
What events can students experience to make the ideas and issues real? What learning activities will help student to explore the big ideas and essential questions?

- Engage students in the discussion of program needs and design.
- Have students create a flowchart for a story problem. Have students create a TOE chart for a story problem.
- Have students sketch two different user interfaces for the program.
- Have students create and code the program.

Reflection/Assessment:
How will you cause students to reflect and rethink to dig deeper into core ideas?
How will you guide students in rehearsing, revising, and refining their work based on feedback and self-assessment?
How will students exhibit their understanding about their final performances and products?
How will you guide them in self-evaluation to identify the strengths and weaknesses in their work and set future goals?

The planning of a program can save a programmer a lot of time. It is important that we first break down what the program needs to do and then what information we need to gather. Using flowcharts and TOE charts are a great way to take the information and prepare it before you create your program.

Think of a problem you need have solved and create a program to do so. Is this a program you would want on your computer, PDA, or phone?